

INET - Sockets in LINUX (1)

• Allgemeines

Sockets der Protokoll-Familie **PF_INET** (INET-Sockets) ermöglichen eine Netzwerkkommunikation unter Verwendung der im **Internet** eingesetzten TCP/IP-Protokolle (Internet-Protokoll Version 4, das neue Internet-Protokoll Version 6 wird durch Sockets der Protokoll-Familie **PF_INET6** realisiert).

Gültige Angaben für den Protokoll-Typ (Parameter *type* der API-Funktion `socket()`) sind:

- **SOCK_STREAM** (→ TCP-Socket)
- **SOCK_DGRAM** (→ UDP-Socket)
- **SOCK_RAW** (→ Raw-Socket)

Zulässige Werte für den Parameter *protocol* der API-Funktion `socket()` (definiert in `<netinet/in.h>`) sind:

- TCP-Sockets : **0** oder **IPPROTO_TCP** (def. in `<netinet/in.h>`)
- UDP-Sockets : **0** oder **IPPROTO_UDP** (def. in `<netinet/in.h>`)
- Raw-Sockets : eine der in RFC1700 den definierten IANA IP-Protokollen zugeordnete Protokoll-Nr.

Eine lokale Adresse, die an einen TCP-Socket gebunden war, ist nach dem Schließen des Socket-Deskriptors für eine gewisse Zeit für eine erneute Bindung nicht verfügbar (es sei denn die Option `SA_REUSEADDR` ist gesetzt gewesen)

• INET-Adressen (IP-Socket-Adressen)

Die Adressen, an die INET-Sockets gebunden werden (Adreß-Familie **AF_INET**), bestehen aus

- einer **IP-Adresse** (Adressen des Rechners im Netzwerk)
- einer **Port-Nummer** (eindeutige Identifikation einer auf dem Rechner laufenden TCP/IP-Applikation)

IP-Adressen sind -im gesamten Netzwerk eindeutige- **32-Bit-Werte** (4 Bytes).

Üblicherweise werden die 4 Bytes durch je einen Punkt getrennt in Dezimaldarstellung angegeben (*Dotted DecimalNotation*): **aaa.bbb.ccc.ddd**

Beispiel: 192.168.206.52

Portnummern liegen im Bereich **0 ... 65535 (16-Bit-Werte)**.

Die Portnummern **0 ... 1023** sind **reserviert** für Prozesse mit Root-Rechten. Hierzu gehören i.a. die "Internet-Standard-Dienste", wie z.B. ftp (Nr. 21), www (Nr. 80) usw. (s. Datei `"/etc/services"`).

Eine **TCP/IP-Verbindung** wird durch **zwei Endpunkte** festgelegt, die **jeweils** durch eine **IP-Adresse** und eine **Port-Nummer** gekennzeichnet sind.

• Structure-Datentyp `struct sockaddr_in`

Dieser in `<netinet/in.h>` definierte Datentyp dient zur Darstellung von **INET-Adressen**.

```
struct in_addr {
    unsigned int s_addr;
} /* Typ zur Darstellung von IP-Adressen */
/* in Network Byte Order */

struct sockaddr_in {
    unsigned short sin_family; /* Address family, hier AF_INET */
    unsigned short sin_port; /* Port-Nr. (in Network Byte Order) */
    struct in_addr sin_addr; /* IP-Adresse (in Network Byte Order) */
};
```

Für **Server-Prozesse** ist i.a. die **lokale IP-Adresse uninteressant** (das bedeutet, sie nehmen Verbindungswünsche auf jeder IP-Adresse, die der Rechner hat, entgegen)

→ für die Komponente `sin_addr` können dann (bei `bind()`) **00-Bytes** angegeben werden ("*don't care*").

Die Angabe von **0** für `sin_port` (Port-Nr. 0) bewirkt, daß der Server auf **jedem Port** auf Verbindungswünsche "lauscht".

INET - Sockets in LINUX (2)

• Darstellungsformate von IP-Adressen und Port-Nummern

IP-Socket-Adressen (IP-Adressen und Port-Nummern) werden über das Netzwerk übertragen. Da Sende- und Empfangs-Rechner unterschiedliche Darstellungsformate für Mehrbyte-Werte haben können (*Little Endian*, *Big Endian*), hat man ein **einheitliches Übertragungsformat** für Mehrbyte-Protokollinformationen festgelegt: **Network Byte Order** (*Big Endian*). (*Big Endian* : das höchstwertigste Byte wird an der niedrigsten Adresse abgelegt bzw als erstes übertragen).

→ Sowohl IP-Adressen als auch Port-Nummern müssen gegebenenfalls zwischen der vom jeweiligen Rechner verwendeten Darstellungsart (*Host Byte Order*) und der **Network Byte Order umgewandelt** werden.

Um dies transparent – ohne Kenntnis der jeweiligen *Host Byte Order* – durchführen zu können, stehen in der **C-Bibliothek geeignete Funktionen** zur Verfügung.

• C-Bibliotheksfunktionen zur Konvertierung zwischen *Network Byte Order* und *Host Byte Order*

Alle Funktionen sind in der Headerdatei `<netinet/in.h>` deklariert.

Umwandlung eines **32-Bit-Wertes** (in LINUX `int == long`) **von Host Byte Order in Network Byte Order**

```
unsigned int htonl(unsigned int hostlong);
```

Parameter : *hostlong* umzuwandelnder 32-Bit-Wert in *Host Byte Order*

Funktionswert : Umgewandelter 32-Bit-Wert in *Network Byte Order*

Umwandlung eines **16-Bit-Wertes** (in LINUX `short`) **von Host Byte Order in Network Byte Order**

```
unsigned short htons(unsigned short hostshort);
```

Parameter : *hostshort* umzuwandelnder 16-Bit-Wert in *Host Byte Order*

Funktionswert : Umgewandelter 16-Bit-Wert in *Network Byte Order*

Umwandlung eines **32-Bit-Wertes** (in LINUX `int == long`) **von Network Byte Order in Host Byte Order**

```
unsigned int ntohl(unsigned int netlong);
```

Parameter : *netlong* umzuwandelnder 32-Bit-Wert in *Network Byte Order*

Funktionswert : Umgewandelter 32-Bit-Wert in *Host Byte Order*

Umwandlung eines **16-Bit-Wertes** (in LINUX `short`) **von Network Byte Order in Host Byte Order**

```
unsigned short ntohs(unsigned short netshort);
```

Parameter : *netshort* umzuwandelnder 16-Bit-Wert in *Network Byte Order*

Funktionswert : Umgewandelter 16-Bit-Wert in *Host Byte Order*

INET - Sockets in LINUX (3)

• C-Funktionen zur Umwandlung der Darstellungsart von IP-Adressen

IP-Adressen werden häufig als Strings in *Dotted Decimal Notation* angegeben.

Andererseits werden sie zur Verwendung für die Socket-API-Funktionen in der `struct in_addr` in **interner Binärdarstellung** (32-Bit-Wert) benötigt.

Zur Umwandlung zwischen diesen beiden Darstellungsarten stellt die **C-Bibliothek** geeignete Umwandlungsfunktionen zur Verfügung.

Diese Umwandlungsfunktionen sind in der Headerdatei `<arpa/inet.h>` deklariert.

Umwandlung einer IP-Adresse aus der String-Darstellung (*Dotted Decimal Notation*) in die 32-Bit-Binärdarstellung

```
int inet_aton(const char* cp, struct in_addr* inp);
```

Parameter : *cp* Pointer auf IP-Adresse in String-Darstellung
inp Pointer auf Struktur-Variable, in der die umgewandelte Binärdarstellung (in *Network Byte Order*) abgelegt wird.

Funktionswert : **!= 0**, wenn durch *cp* eine gültige IP-Adresse referiert wird
0, wenn keine gültige IP-Adresse referiert wird

Umwandlung einer IP-Adresse aus der 32-Bit-Binärdarstellung in die String-Darstellung (*Dotted Decimal Notation*)

```
char* inet_ntoa(struct in_addr in);
```

Parameter : *in* IP-Adresse in Binärdarstellung (*Network Byte Order*)

Funktionswert : Pointer auf einen String, der die IP-Adresse in *Dotted Decimal Notation* enthält.

Anmerkung : Der zurückgegebene String befindet sich in einem statisch allozierten Buffer, der beim nächsten Aufruf der Funktion überschrieben wird.